# Autonomy and Software Technology on NASA's Deep Space One

R. Doyle, D. Bernard, E. Riedel
N. Rouquette, J. Wyatt
*Jet Propulsion Laboratory*
*Pasadena, CA*

M. Lowry
P. Nayak
*NASA Ames Research Center*
*Moffett Field, CA*

**(IEEE Intelligent Systems
May/June 1999)**

NASA's Deep Space One (DS1) mission is unprecedented. Normally justified by science data return as the primary, if not the sole consideration, DS1 is the first NASA mission whose main purpose is to demonstrate the flight readiness of a set of technologies. DS1 is the vanguard of NASA's New Millennium Program, which was conceived to directly address the ongoing challenge of flight qualifying technologies for mission use, and to short-circuit the Catch-22 situation where flight project managers naturally prefer to utilize technologies only after they've been flown on another mission. Any of the DS1-qualified technologies may hold the key towards enabling future NASA space exploration missions. DS1 carries a dozen technology experiments, each demonstrating new capabilities which cover the gamut of spacecraft functions from propulsion to sensing to power to communications. Three of these technology experiments demonstrate new capabilities in spacecraft autonomy and autonomous mission operations.

The autonomy-related experiment on DS1 with the largest scope is a joint NASA Ames Research Center (ARC) / Jet Propulsion Laboratory (JPL) project known as the **Remote Agent** (RA). RA is both an autonomy architecture and a set of component reasoning engines for the functions of mission planning, execution and fault protection.

One of the fundamental space mission functions is navigation. DS1 is the first interplanetary mission to be navigated by an **Autonomous Navigation** system of any type. All previous missions have been navigated by ground operators. The DS1 navigation technology demonstration (AutoNav) breaks this ground-link, and enables a spacecraft to navigate independently of ground teams and ground-links.

As spacecraft begin to become more autonomous, mission operations concepts must also evolve. The **Beacon Operations** (BMOX) experiment on DS1 demonstrates a new end-to-end concept for mission operations, where the spacecraft takes responsibility for determining when ground support is needed.

One of the lessons learned by the NASA autonomy technologists is that software engineering issues emerge as on the critical path towards realizing autonomy capabilities. One of the most daunting autonomy software issues is that of testing. Encouragingly, a technique based in formal methods yielded important results on DS1 when an error was detected early on in a component of the Remote Agent.

Another technique in software engineering yielded crucial results on DS1 when automatic code generation was successfully used to generate much of the core fault protection flight code. Although not an official technology experiment, this success nonetheless enabled DS1 to meet its intense development schedule and launch in October 1998.

The Remote Agent team is being led by Doug Bernard at JPL and by Pandu Nayak at ARC. The Remote Agent Experiment (RAX) is a flight experiment that demonstrates a new approach to spacecraft commanding and control. In the Remote Agent approach, the operational rules and constraints are encoded in the flight software and the software may be considered to be an autonomous "remote agent" of the spacecraft operators in the sense that the operators rely on the agent to achieve particular goals. The operators do not know the exact conditions on the spacecraft, so they do not tell the agent exactly what to do at each instant of time. They do, however, tell the agent exactly which goals to achieve in a period of time as well as how and when to report in. This Remote Agent approach is being designed into the New Millennium Program's Deep Space One (DS1) mission as an experiment.

The DS1 Remote Agent Experiment has multiple goals. A primary goal of the experiment is to provide an on-board demonstration of spacecraft autonomy. This includes goal-oriented commanding, time-driven and event-driven execution, and model-based fault diagnosis and recovery. An equally important, and complementary, goal of the experiment is to familiarize the spacecraft engineering community with the Remote Agent approach and to decrease the risk (both real and perceived) in deploying Remote Agents on future missions.

The Remote Agent is formed by the integration of three separate Artificial Intelligence technologies: an on-board planner-scheduler, a robust multi-threaded executive, and a model-based fault diagnosis and recovery system. All three are written in Harlequin Lisp specifically ported to run under VxWorks on a RAD6000 processor. Fundamentally, each of the three technologies can be thought of as using two distinct components: a general purpose reasoning engine and application-specific models. The Remote Agent has been designed to operate at several different levels of autonomy ranging from traditional spacecraft commanding through on-board planning and execution. The Remote Agent flight validation proceeded as follows: first the RA was used to handle low-level commands as instructed by the ground. Next, the ability of the RA to execute a flexible plan generated on the ground was demonstrated. Finally, the RA was given approval to generate plans on-board and execute them without prior inspection of those plans by humans. In the course of the experiment, several fictitious failures were injected, giving the RA an opportunity to demonstrate its model-based fault protection approach. With ground and flight testing now complete, all Remote Agent validation objectives have been met and the team is turning its attention to how to apply the lessons learned during the experiment to future technology upgrades.

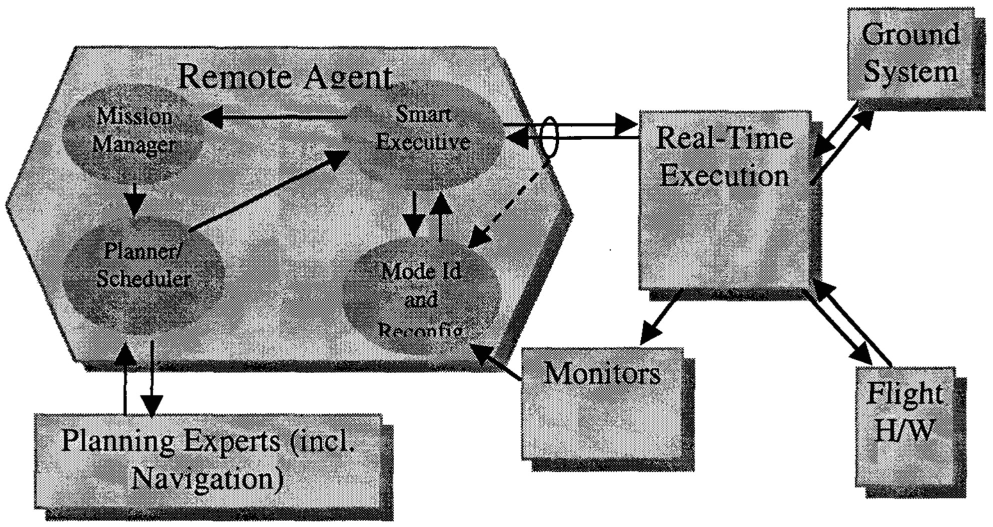


**Figure 1. Remote Agent Architecture.**

The Autonomous Navigation team is being led by Ed Reidel at JPL. The Autonomous Optical Navigation (AutoNav) system, by using images from an onboard camera of sufficient quality AutoNav can control the flight path of a spacecraft, including use of Solar Electric Propulsion (SEP), and target one or more flyby encounters, or rendezvous. The system was designed to be largely self contained, meaning it can be inserted into a fairly simple software architecture, without other autonomous systems required, except for Attitude Control (ACS). This is the situation for DS1. Even in the simple "traditional" environment of the borrowed Mars Pathfinder software set, highly autonomous behavior has been achieved. The advantages of this system, and the success in its utilization so far, have encouraged several missions to baseline its use in whole or part, these include the Space Technology-4 mission to rendezvous and land with a comet, Stardust, which will use the close approach system, and Deep-Impact, a Comet "penetrator" mission, which will use encounter and targeting components of AutoNav.

AutoNav consists of several subsystems and functions:

- Navigation Executive Function - The "NavExec" controls all AutoNav operations that cause physical action by the spacecraft. By Communicating with ACS, NavExec accomplishes the complex of activities necessary to turn the spacecraft and image a series of target navigational "beacons," which in the case of DS1 are usually "nearby" asteroids. These activities include planning the sequence of turns to optimize time utilization and insure completion of the photo-taking sequence on schedule. NavExec also performs similar duties during the long segments of SEP activity, wherein the spacecraft is commanded by NavExec to go to the required attitude, light the engine, and maintain thrust at periodically updated attitudes and magnitudes. Similarly, NavExec commands the execution of Trajectory Correction Maneuvers.

- Image Processing - The Image Processing function, as its name implies, is responsible for identifying the objects and stars in images relayed to AutoNav, and doing highly precise data reductions. Ultimately 0.1 pixel (picture element) accuracy is anticipated from the algorithms (although current scattered-light and other problems with MICAS prevent this from being achieved). Special encounter image processing is included to amplify the dim signal of the target as seen from many hours before closest approach.
- Orbit Determination (OD) - Using data from the Image Processor, AutoNav computes the position of the spacecraft through the use of a batch-sequential modified Kalman filter. Parameters modeling SEP thrust, and random accelerations (e.g. errors in solar pressure modeling, or spacecraft out gassing) are also estimated.
- Maneuver Planning - With the results of the OD in hand, AutoNav will compute updates to the upcoming SEP thrust plan, or the components of a "statistical" trajectory correction maneuver (i.e. one based on statistical variations in the OD). These TCM's can use either SEP or the hydrazine propulsion system.
- Encounter Knowledge Updates: After the final TCM is performed, AutoNav switches to a special mode of activity, which specifically updates onboard knowledge of the target position, and relays this information to ACS for spacecraft pointing changes.

AutoNav began operations as soon as the spacecraft came to life after launch on October 24, 1998, providing critical ephemeris information to ACS. Over the following four months, progressively more components of AutoNav were checked out, and invoked, until April 20, 1999 the spacecraft came completely under the control of the AutoNav system, flying a SEP powered flight path computed onboard. It is anticipated that this autonomous control will continue with periodic necessary suspensions or updates in AutoNav control for onboard tests and validations, leading to a fully autonomously controlled flyby of asteroid 1992KD on July 29, 1999.

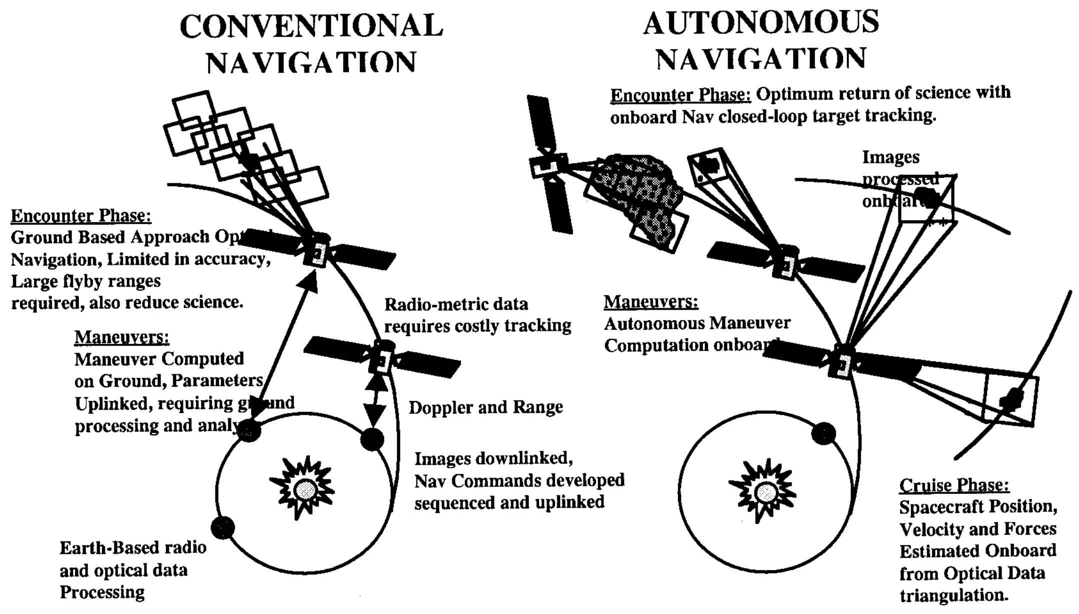

CONVENTIONAL NAVIGATION
AUTONOMOUS NAVIGATION
Encounter Phase: Optimum return of science with onboard Nav closed-loop target tracking.
Images processed onboard
Encounter Phase:
Ground Based Approach Op
Navigation, Limited in accuracy,
Large flyby ranges
required, also reduce science.
Radio-metric data requires costly tracking
Maneuvers:
Autonomous Maneuver Computation onboa
Maneuvers:
Maneuver Computed on Ground, Parameters Uplinked, requiring g und processing and analy
Doppler and Range
Images downlinked, Nav Commands developed sequenced and uplinked
Earth-Based radio and optical data Processing
Cruise Phase:
Spacecraft Position, Velocity and Forces Estimated Onboard from Optical Data triangulation.

**Figure 2. Comparison of Conventional and Autonomous Spacecraft Navigation.**

The Beacon Operations team is being led by Jay Wyatt at JPL. The purpose of this experiment is to flight validate an operations concept and the associated technology components necessary to enable more adaptive operations on future space missions. The value of this approach is tri-fold. First, it enables the spacecraft to ground link to be achieved more cheaply from both a spacecraft resources standpoint as well as a mission operations cost perspective. Second, it reduces the routine tracking burden of large aperture antennas, which can help NASA's Deep Space Network reduce the loading on its overconstrained antenna network. Third, it can reduce mission risk since the low-cost link can be maintained more frequently and/or during times in a mission when the telemetry link cannot be achieved due to spacecraft or mission design constraints.

Two subsystems implement the beacon operations functionality on DS1. The first is an end-to-end tone system that enables the spacecraft to inform the ground whether or not data needs to be sent. This tone does not contain any telemetry but rather represents one of four possible requests for ground action (no action required, contact when convenient, contact within a certain time, or contact immediately). The second subsystem produces intelligent data summaries that are downlinked as telemetry after ground personnel respond to the tone request. Onboard summarization produces four types of engineering telemetry. High-level spacecraft information, such as the number of alarm crossings, spacecraft mode and state histories, and other pertinent statistics are gathered since the last ground contact. Episode data is gathered for the culprit and causally related sensor channels whenever a sensor violates an alarm threshold and is stored at a high sample rate. Snapshot telemetry is collected at a much lower sample rate for all sensor and transform channels. Snapshot data is used only for rough correlation and to fill in the gaps between episodes. The last component of the downlinked summary, Performance Data, is similar to episode data but captures maneuvers or other events that are known in advance to be of interest to people on the ground. All of the summary algorithms are implemented in C for the VxWorks operating system.

The summary algorithms incorporate AI-based methods to enhance anomaly detection and episode identification capability. The ELMER (Envelope Learning and Monitoring using Error Relaxation), technology replaces traditional red-lines with time-varying alarm thresholds to provide faster detection with fewer false alarms. These functions are learned using a neural network and training can be performed onboard or on the ground (ground-based for DS1). ELMER is particularly powerful because very little knowledge engineering is required and training of the neural net is accomplished with nominal sensor data. Another AI-based method produces empirical transforms that derive their heritage from previous AI research work at JPL in the area of selective monitoring. Once computed onboard, these act as pseudo sensors. The current transforms for DS1 compute high, low, and average values, first derivative, and second derivative. Alarm limits can be placed on these transforms and they can also serve as an input to the ELMER neural network. Additional transforms, if desired, can easily be defined and uplinked to the spacecraft as the mission progresses.

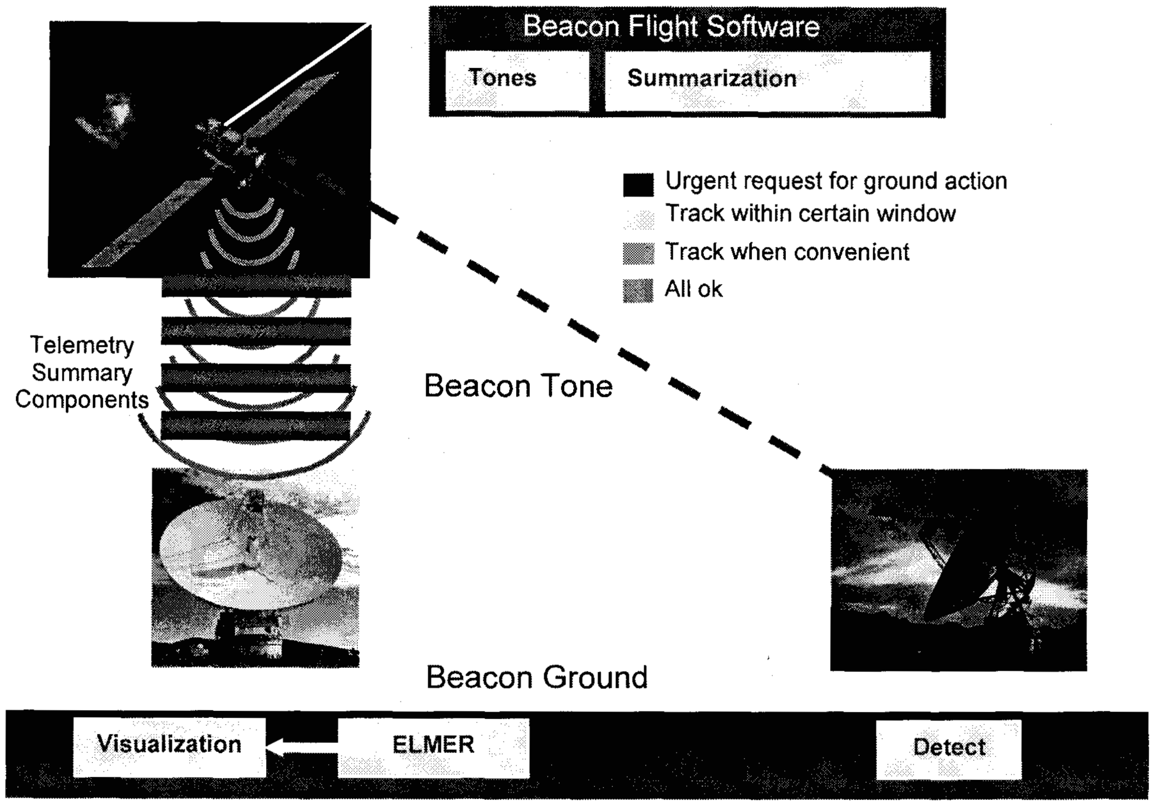


**Figure 3. Beacon Operations System**

The DS1 autonomy software testing work was led Mike Lowry at ARC. Within the realm of space exploration a major obstacle to widespread application of autonomy capabilities in flight software is not just technical feasibility; it is doubt about its trustworthiness as a replacement for human-in-the-loop decision-making. Autonomous control systems raise difficult verification & validation issues because, unlike conventional sequencer-based open-loop systems that perform transactions visible through uplink/downlink communications, they close many control loops and arbitrate many resources onboard with specialized reasoning in multiple concurrent threads. The number of possible execution paths for autonomous control systems is many orders of magnitude greater than traditional flight control software. What is needed are V&V techniques that significantly increase confidence in these decision-making control systems. Towards this end, researchers at NASA Ames, JPL, and Carnegie Mellon University demonstrated techniques for verifying the greatly expanded number of possible execution paths inherent in autonomy software. Techniques were demonstrated for all parts of the remote agent – planner, executive, and MIR- with the most extensive demonstration focusing on the resource manager of the goal-oriented executive.

Autonomy software is inherently concurrent - that is, multiple processes achieving different goals or sub-goals are executed in parallel. Concurrent task software is easier to program than traditional sequences because the means of achieving each goal can be designed separately. Because of the closed-loop nature of autonomy, each goal being achieved represents a separate process. However, the extra degrees of freedom in achieving goals through separate processes can lead to unintended interactions between processes and lead to failures. It is these extra degrees of freedom that make autonomy software verification difficult through testing alone.

The technology of model-checking has been previously used to debug and verify concurrent digital hardware designs and communication protocols. Most of the demonstrations focused on using model-checking to debug and verify portions of the remote agent. Model-checking is a set of mathematical algorithms based on automata theory for verifying and debugging concurrent or real-time systems modeled as interacting finite state machines. Given a model and a property, a model-checker searches for *traces* of the model that violate the property -a trace is an interleaved sequence of states of the finite state machines. Model checkers differ from simulators in that they explore all relevant traces. In other words, they explore all realizable paths through the graph of states that can be reached from the initial state and that match the property being checked. Model checkers are particularly well suited to exploring the relevant execution paths of non-deterministic systems with multiple processes running in parallel. This makes them well adapted to verification and debugging of autonomy software.

For the procedural executive, model-checking was applied directly to the core software. Fivc concurrency bugs were found by using model-checking to explore all possible execution traces. The errors found were for unusual situations that were not fully considered by the designers. For example, a race condition was found for the situation where a task program was aborted and the locks it had on resources were not released correctly if the demon monitoring the locks woke up at a particular point. For the planner and MIR, both of which are based on deductive methods as opposed to procedural methods, model-checking was applied to validate the models. Specifically, it was demonstrated how model-checking could be used to find possibly unintended consequences of a model, and thus help the model developer revise the model. Finally, it was shown how run-time verification could be tied into the same framework as model-checking through *behavior auditors* that monitor the run-time execution of autonomy software. Behavior auditors are specified in a language similar to the property descriptions used by model-checkers.

The before and after graphic in Figure 4 illustrates a critical step in current approaches to analytic verification - namely, manually abstracting a software system so that current verification algorithms can detect bugs. This particular graphic illustrates the abstraction of the remote agent resource manager, in which five anomalies were detected through analytic verification algorithms.

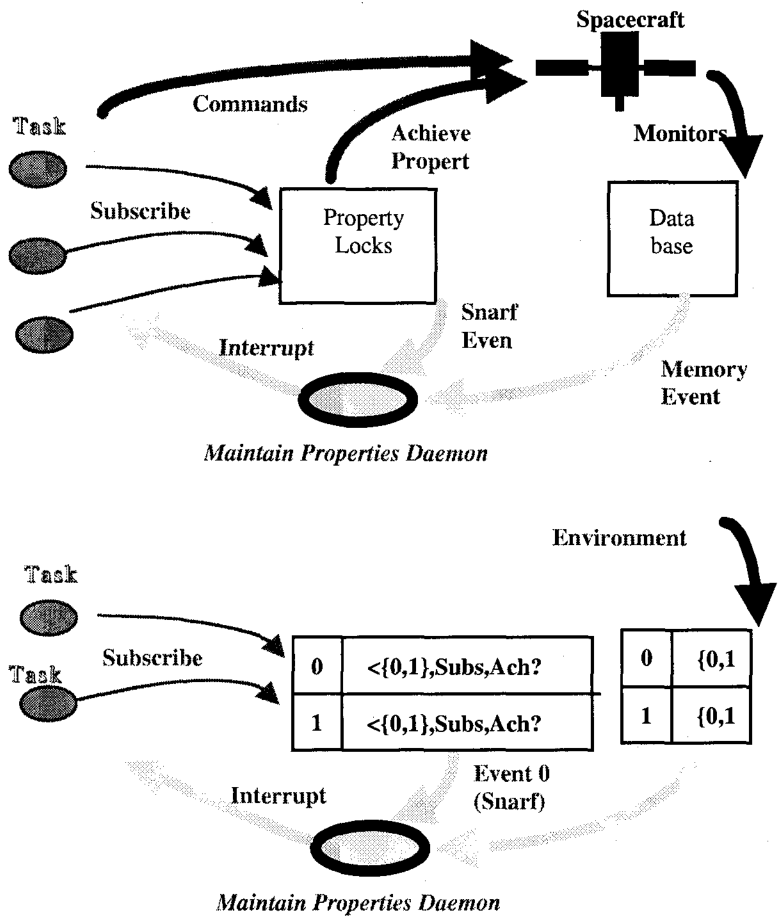


**Figure 4. Software Abstraction of Remote Agent Resource Manager**

The auto-coding work on DS1 was led by Nicolas Rouquette at JPL. Until DS1, the JPL had not used code-generation techniques on large scale for avionics software. However, the constraints of the mission design and development cycle, limited budget and resources dictated a departure from past practices. The demands of concurrent design and development as well as overlapping design and integration schedules would have drained all available resources allocated for system-level fault-protection. The requirement that post-launch activities be directed by fault protection further increased the difficulty of the task; on other spacecraft, such activities are typically handled with sequences.

First, the DS1 project decided to reuse the successful Mars Pathfinder (MPF) fault-protection engine as it achieves a good separation of architecture and domain concerns. However, MPF relied heavily on software engineers to encode the domain-specific fault-protection design into the target C language. Despite fewer software-engineering resources, DS1 faced a nonetheless larger design scope (spinner vs. 3-axis spacecraft) that also included post-launch activities that, on other spacecraft, are typically handled with sequences. To step up to this challenge, the fault-protection strategy was reorganized to capture all designs in terms of high-level behavioral and structural specifications instead of low-level C code. The Harel statechart notation became the standard means of describing the design of every fault protection monitor and response. Such monitors are responsible for extracting features from raw data to reliably detect the occurrence of a known fault while fault responses define the logic that controls the spacecraft to mitigate the effects of a fault.

DS1 represents the first spacecraft at JPL where code generation from statecharts has been systematically applied to achieve rigorous and consistent software implementations in the target language directly from the statechart diagrams. This process was supported, in part, with the Mathwork's Matlab Stateflow' toolbox. This toolbox provides a customizable translative code generator for statecharts. Extensive customizations of that toolbox were necessary, first to address the needs of the DS1 fault-protection runtime architecture and second, to fulfill the end-to-end needs of the mission from design, to software, integration and test and to operations.

To support the systematic comparison of test results obtained from any pair of platforms (unit test, testbeds, and spacecraft), the execution of fault responses had to be adequately instrumented. Knowledge of the set of all fault and responses was leveraged to derive a minimal-length encoding/decoding algorithm for on-board compression (encoding) and ground decompression (decoding) of state transition events. Instead of instrumenting the state transition code of each fault response, the statechart execution architecture is instead instrumented to signal state-transition events to the compression algorithm. From a sequence of such events, this algorithm computes an encoded value representing the path to the current state from the top-level statechart (i.e., the fault response handling the occurrence of a fault) through all intermediate statecharts invoked (i.e., the hierarchy of helper sub-responses invoked). The encoded path and ancillary sampled variables constitutes an event record. These algorithms produce the necessary and sufficient set of event records to provide full accountability of the rationale behind every state transition for every fault response execution within memory limitations. This technique enables a behavior reconstruction approach to testing, the process of producing a parsimonious explanation of a fault-response execution by determining the sequence of external events that recreates the same event record history as that obtained from the spacecraft.

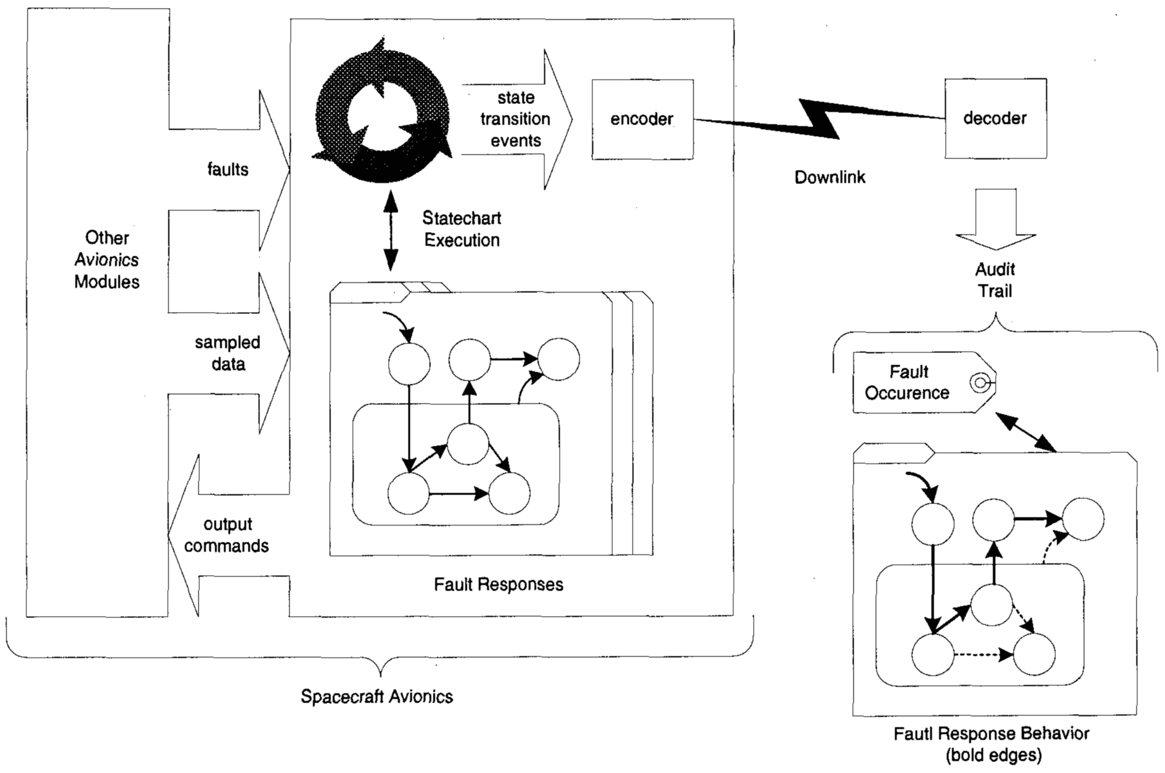
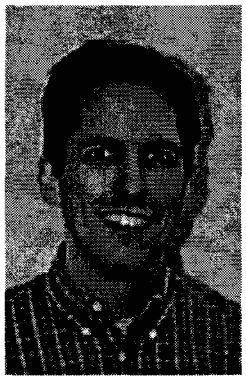


**Figure 5. Behavior Reconstruction in Auto-coding.**

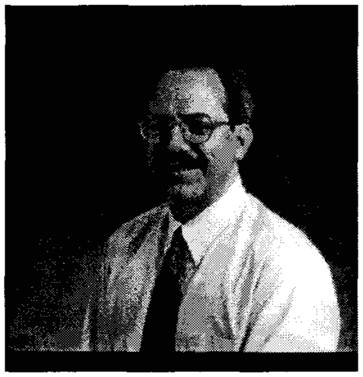
These five autonomy technology experiments and related software engineering activities on DS1 are paving the way for the use of autonomy capabilities in future NASA missions: proving technologies, reducing perceived risk, and ameliorating first user costs. NASA is entering the era of autonomous space systems, and the results achieved on DS1 are already leading to applications of the autonomy technologies described here as well as inspiring additional autonomy technology development work.

Portions of the research described in this article were performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

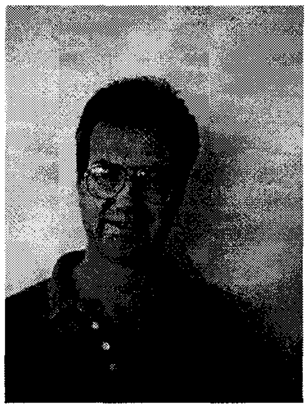
**Dr. Douglas Bernard** is the Technical Group Supervisor for the Avionics Flight System Engineering group at JPL and team lead for Remote Agent autonomy technology development for the New Millennium Program. He received a B.S. in Mechanical Engineering and Mathematics from the University of Vermont, a M.S. in Mechanical Engineering from MIT and a Ph.D. in Aeronautics and Astronautics from Stanford University. He has participated in dynamics analysis and attitude control system design for JPL spacecraft including the Galileo mission to Jupiter and the Cassini mission to Saturn.



**Dr. Richard Doyle** is Technical Division Manager of the Information Technologies and Software Systems Division and Leader of the Center for Space Mission Information and Software Systems at JPL. He formerly held the roles of Technical Section Manager of the Information and Computing Technologies Research Section and Program Manager for Autonomy Technology at JPL. He received his Ph.D. in Computer Science at the MIT Artificial Intelligence Laboratory in 1988. He is a Technical Program Chair for the 5th International Symposium on Artificial Intelligence, Robotics and Automation for Space, at Noördvijk, The Netherlands in June 1999. He gave the Invited Talk entitled "The Emergence of Spacecraft Autonomy" at the National Conference on Artificial Intelligence in Providence, RI in July 1997.

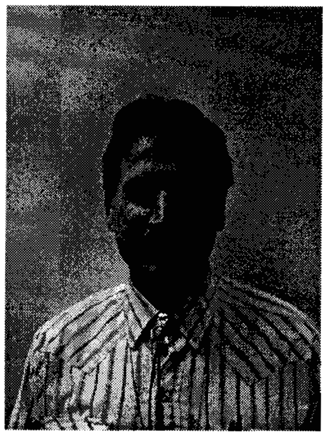


**Dr. Michael Lowry's** research interests are in automated high-assurance methods for software generation and software verification. After receiving his PhD from Stanford University, he edited the MIT press book "Automating Software Design" (1991) and has served on the editorial board of the Kluwer journal 'Automated Software Engineering' since its beginning. After joining NASA Ames research center in 1992, he started the Amphion project, which has developed technology that concurrently generates software programs and proofs of correctness. The software programs are compositions of software components, typically from NASA libraries. In 1996 he started research on the verification of autonomous systems. Dr. Lowry currently leads the Automated Software Engineering group at NASA Ames, which includes over a dozen PhD-level researchers with a broad international background.

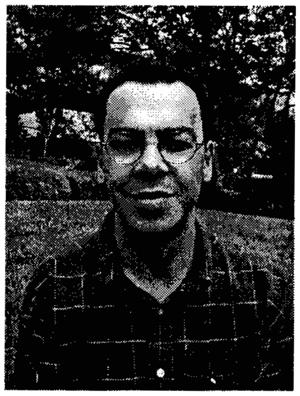


**Dr. Pandurang Nayak** is a research scientist with RIACS at the NASA Ames Research Center, and a consulting faculty at Stanford University. He holds a Ph.D. in computer science from Stanford University (1992), and his dissertation was an ACM Distinguished Thesis. He is currently an associate editor of JAIR, and his research interests include model-based autonomous systems, diagnosis and recovery, abstractions, qualitative and causal reasoning. His work on incremental truth maintenance won a best paper award at AAAI-97.

**Ed Riedel** joined the Voyager Mission Navigation team at JPL in 1978. He participated in the six planetary encounters of both spacecraft, and was the lead optical navigation engineer for the Voyager 2 Neptune flyby. After leading the optical navigation team for the Galileo asteroid flybys and early Galilean tour, he became the New Millennium DS1 Navigation Team chief, directing the development of a completely autonomous optical navigation system to guide the approach and flyby of DS1 past an asteroid and two comets. He is an author of numerous papers on navigation, optical navigation and navigation-related image processing technology.



**Dr. Nicolas Rouquette** is a Senior Computer Scientist at the Jet Propulsion Laboratory. He holds an Electrical Engineering degree from ESIEE in France and a Masters and Ph.D in Computer Science from the University of Southern California. He was the software architect and engineer of the DS1 fault-protection software. His research interests are in Artificial Intelligence, Model-Based Reasoning, Computational Mathematics, and Software Engineering. His current work focuses on deploying statechart technology in embedded systems.



**Jay Wyatt** is the principal investigator for the Beacon Monitor Operations Experiment on DS1. He also manages the Infrastructure and Automation work area at JPL, which funds research and development in adaptive mission operations, advanced telecommunications protocols, and automation of the Deep Space Network. Jay is also the technical group supervisor for a mission autonomy and systems engineering group within the JPL Information & Computing Technology Research Section. His current research interests are in operations concepts for missions using autonomy, and more generally, in monitoring and diagnosis of complex systems. Prior to Deep Space One, Jay led software design & development activities for the Pluto and Europa missions. Jay has approximately ten years of experience working within the NASA community, three of which were spent developing space station life support systems at the Marshall Space Flight Center.